

中山大學

SUN YAT-SEN UNIVERSITY

# 本科生实验报告

实验课程：操作系统原理实验

实验名称：添加新的系统调用

专业名称：计算机科学与技术

学生姓名：张玉瑶

学生学号：23336302

实验地点：

实验成绩：

报告时间：2025 年 3 月 27 日

## Section 1 实验概述

本次实验学习在 linux 内核中添加新的系统调用，以便我们更加深入地理解内核的工作原理。

## Section 2 实验步骤与实验结果

### 1. 实现系统调用函数

在 linux5.10.235 文件中打开 kernel 文件，创建 hello.c 文件后输入如下代码。

```
// kernel/hello.c
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(hello) {
    printk(KERN_INFO "Hello from syscall!,23336316\n");
    return 0;
}
```

### 2. 添加系统调用号

依照 arch/x86/entry/syscalls/syscall\_64.tbl，进入文件在最后一行添加如下语句。

```
408 # This is the end of the legacy x32 range. Numbers 548 and a
409 # not special and are not to be used for x32-specific syscall
410 588      common hello      sys_hello
```

3. 修改 linux5.10.235/kernel 中的 Makefile 文件，添加如下语句。（注意不是外面的 kernel!）

```
13      async.o range.o smpboot.o ucount.o regs
14 obj-y +=hello.o
```

### 4. 修改系统调用头文件

在 include/linux/syscalls.h 中文件末尾、#end if 之前添加如下语句。

```
1363 asmlinkage long sys_hello(void);
1364 #endif
```

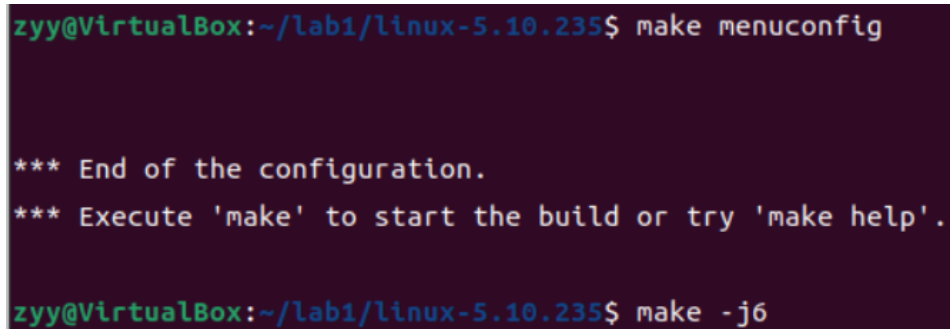
### 5. 增加测试文件

在 lab1 创建文件 test\_syscall.c，输入如下代码。

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/syscall.h>
4 #include <errno.h>
5 #include <string.h>
6
7 int main() {
8     long ret =syscall(588);
9     printf("return code is %ld\n",ret);
10
11     return 0;
12 }
```

## 6. 编译内核

在 linux-5.10.235 的终端中输入以下代码。



```
zyy@VirtualBox:~/lab1/linux-5.10.235$ make menuconfig

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

zyy@VirtualBox:~/lab1/linux-5.10.235$ make -j6
```

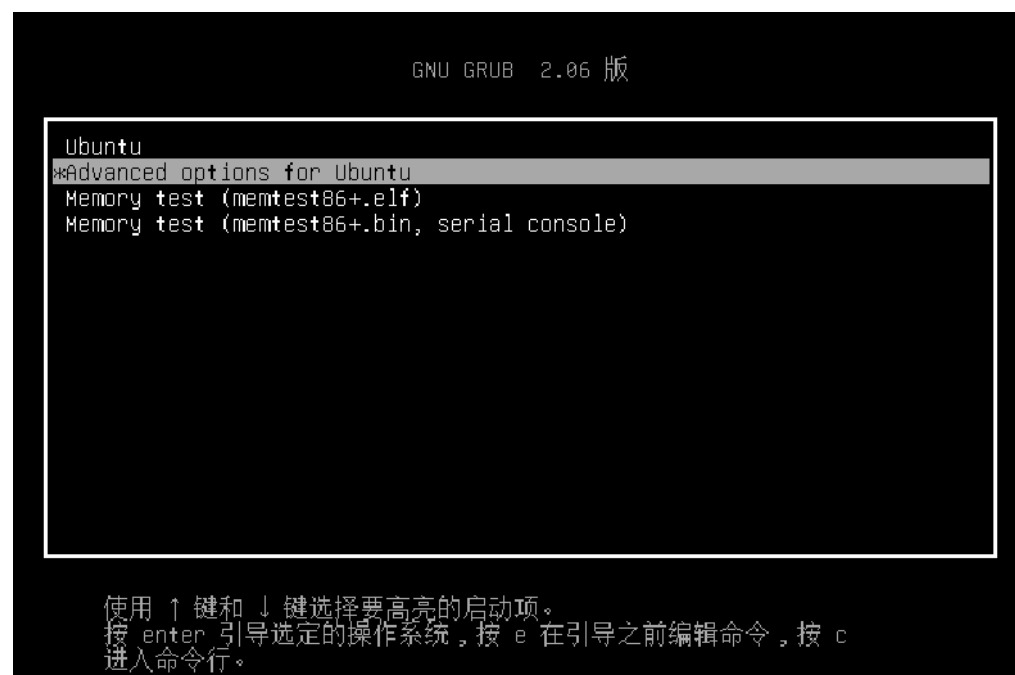
最后可以看到语句 Kernel: arch/x86/boot/bzImage is ready , 说明编译成功!

```
AS      arch/x86/boot/compressed/elf_chunk_01.o
CC      arch/x86/boot/compressed/misc.o
GZIP    arch/x86/boot/compressed/vmlinux.bin.gz
MKPIGGY arch/x86/boot/compressed/piggy.S
AS      arch/x86/boot/compressed/piggy.o
LD      arch/x86/boot/compressed/vmlinux
ZOFFSET arch/x86/boot/zoffset.h
OBJCOPY arch/x86/boot/vmlinux.bin
AS      arch/x86/boot/header.o
LD      arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready (#3)
zyy@VirtualBox:~/lab1/linux-5.10.235$
```

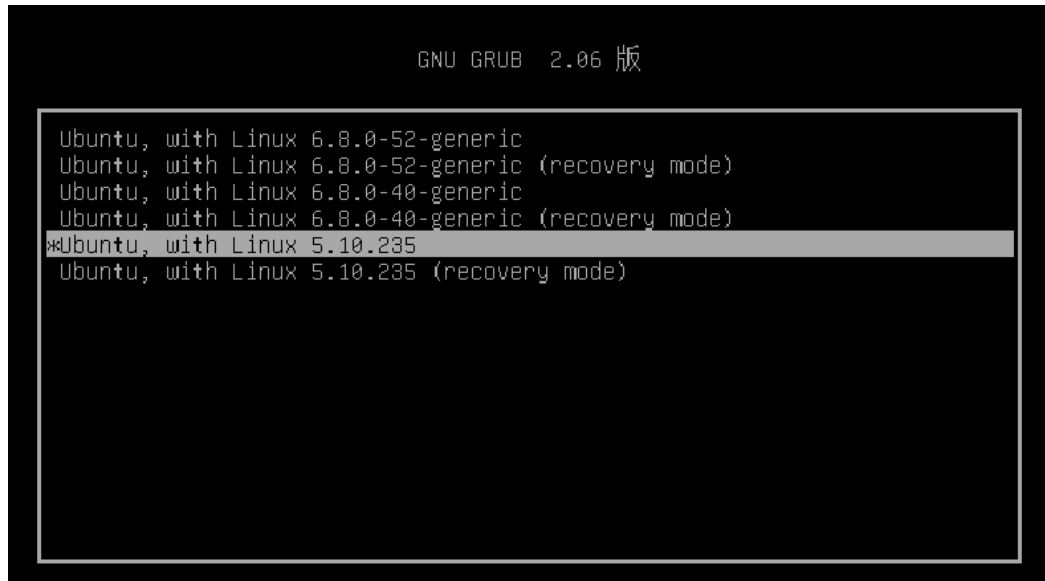
## 7. 安装内核

```
sudo make modules_install install
```

重启，重启时长按 shift 进入选择项选择 advanced options for ubuntu 选项。



进入如下内核。



在 lab1 终端中编译 test\_syscall.c 文件，再进行测试。可以看到 return 了 0。

```
zyy@VirtualBox:~$ cd lab1
zyy@VirtualBox:~/lab1$ gcc -o test_syscall.o ./test_syscall.c
zyy@VirtualBox:~/lab1$ ./test_syscall.o
return code is 0
```

最后输入 dmesg，返回如下，成功执行了新的系统调用！

```
[ 175.675628] gsd-power[2037]: segfault at 8 ip 00007f45c0edeb40 sp 00007ffd6e
24eea0 error 4 in libupower-glib.so.3.1.0[7f45c0ed1000+13000]
[ 175.675637] Code: 8b 3c 24 ba 13 00 00 00 89 c6 e8 fb 36 ff ff 85 c0 75 bb 4
8 8b 04 24 48 8d 15 cc 67 00 00 be 10 00 00 00 48 8d 3d 4a 55 00 00 <48> 8b 48
08 31 c0 e8 a5 3b ff ff eb 97 e8 fe 37 ff ff 66 66 2e 0f
[ 219.329333] Hello from syscall!,23336316
```

## Section 3 实验总结与心得体会

这次增加系统调用的实验看似简单，步骤也少，但是还是吃了很多亏。第一次吃亏是没有分辨清楚不同文件下的 Makefile，改错了导致我重新搞了一个内核，很崩溃。第二次是不知道为什么编译总要一两个小时还经常报错，忍无可忍，直到现在我都不知道错在哪。总之过程非常折磨，非常搞我心态，但看到成功调用出来“Hello from syscall!, 23336316”的时候还是挺开心的。